Computing Information by Equation Solving

Kiyong Lee
Korea University

0. Introduction

Assuming that representation is a crucial issue for Computational Semantics, I will show in this paper that equation solving is a very simple and elegant way or computing meaning representations or information structures conveyed by the use of natural language. In a formal semantics like Montague's(1974), a type-theoretic λ -calculus with β-reduction is often used to handle with substitutions. But I argue in this paper that the substitution for equation solving is less constrained that the substitution by β-reduction and that the former can treat the phenomena of free word-order and partial information in natural language more conveniently that the latter. To support my argument, I will analyze some fagments of Korean and English, particulary those involving scrambling in Korea, and quantification and NP-conjunction in English.

1. Two Types of Substitution

There are two types of substitution in computing meaning representations: one by β -reduction and the other, by equation solving. Consider the following substitutions instances:

(1) β-reduction
  a. $\lambda x \lambda y f(x,y)(b)(a)$
  b. $\lambda y f(a)(b)$
  c. $f(a,b)$
(2) equation solving
  a. $f(x,y)$, $x=a$, $y=b$.
  b. $f(a,y)$, $y=b$. OR b'. $f(x,b)$, $x=a$.
  c. $f(a,b)$.

In (1), the functor $\lambda x \lambda y f(x,y)$ takes two arguments *a* and *b*, yielding a closed formula $\lambda x \lambda y f(x,y)(b)(a)$. This formula is reducible to $f(a,b)$ by the repreated application of β-reduction: first, *a* substitutes for *x* and then *b* for *y*. Unlike the formula (1a), the formula $\lambda x \lambda y f(x,y)(a)(b)$ reduces to a different formula $f(b,a)$ by substituting *b* for *x* and *a* for *y*. But in the case (2) of equation solving, there is no fixed order of substitution: whatever the order of application may be, *a* always substitutes for *x* and *b* for *y*. (2a) is thus equivalent to $f(x,y)\{x=a,\ y=b\}$, where there is no constraint over the order of substituting *a* for *x* and *b* for *y*.

Language, Information and Computation
Chungmin Lee and Beom-mo Kang, editors, 1993, Seoul: Thaehaksa

There is another difference between β-reduction and equation solving. In the β-reduction approach, the meaning of a sentence is normally represented by a single formula, and this formula may often consist of an *n*-ary functor and its *n* arguments. This complex formula is then reduced by β-reduction to a simpler or atomic formula. In the equation-solving approach, on the other hand, the meaning of a sentence may be represented by a set of open formulas and these formulas may reduce to a single formula by the process of equation solving. In the former, a formula consisting of a functor and its arguments is not decomposed into other formulas, whereas in the latter a set of formulas may decompose to simpler formulas or simpler sets of formulas. If we accept the view that atomic formulas represent basic units of factual or non-factual information, then the equation-solving approach can represent any partial information associated with parts of a sentence in a more elegant way as will be illustrated in this paper.

2. Scrambling in Korean

In Korean, no-verbal expressions like NP's can be scrambled in a sentence, as in:

(3) a. Kim-i Ann-eke ton-ɨl cuəssta.
Kim-nom Ann-dat ton-acc gave
'Kim gave Ann money.'
b. Kim-i ton-ɨl Ann-eke cuəssta.

c. Ann-eke Kim-i ton-ɨl cuəssta.
d. Ann-eke ton-ɨl Kim-i cuəssta.
e. ton-ɨl Kim-i Ann-eke cuəssta.
f. ton-ɨl Ann-eke Kim-i cuəssta.

Despite a difference in word order, they all have the same semantic representation.

(4) cuəsssta(kim,ann,ton) 'Kim gave Ann money'

In the traditional Montague grammar based on λ-calculus, we had to set several different categories of the di-transitive verb cu-ta 'give' and several different rules of combination. But in the proposed equation-solving approach, we need only a single recursive PS rule for the category V and an NP rule introducing a CM, Case Marker:

(5) Rules

V --> NP V
NP --> N CM

We also need a Lexicon in which each lexical item is provided with appropriate information or necessary constraints. Factual information is mostly represented in an equational form, while a constraint is represented by a rule form *p* :- *q* read '*p* if *q*', where *p* is a head and *q*, a body[1].

Before I discuss scrambled sentences in (3), I will analyze a simpler sentence **kim-i** conta 'Kim dozes' for illustration's sake.

(6) Lexicon

conta: V, v(SEM)=[conta(X) :- member(X,Agent)], Role=Agent :- Case=nom.

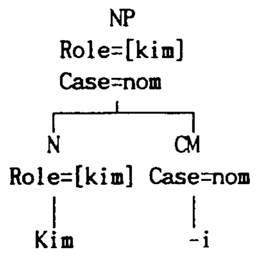
Kim: N, Role=[kim].

-i: CM, Case=nom.

Here, conta 'doze' belongs to a category V and its semantic content is a constraint or rule consisting of a head conta(X) and a body member(X,Agent), where Agent is a list of agents.[2] This verb contains a constraint that relates the nom(inative) Case to the argument Role Agent. The noun **kim** denotes a list [kim] with an uninstantiated Role: this Role will be instantiated when the noun combines with an appropriate verb. The Case Marker(CM) -i carries information about the Case-value nom.

Given the Lexicon, the Case-Marking rule licenses the following tree:

(7)



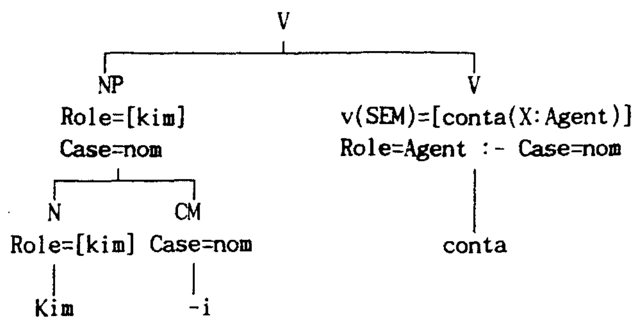
Now, by unifying pieces of information given in the daughter nodes N and CM,[3] we obtain the meaning or information structure associated with the mother node NP as a list of equations [Role=[kim],Case=nom].

The V Rule will then combine the NP structure given above with V, yielding the following:

1 In this paper, I will freely use terminology or notations commonly used in Prolog.
2 Here, I will introduce an infix ':', as defined: *X:L :- member(X,L)*. By using this infix, we can treat a formula *f(X:L)* as equivalent to *f(X) & member(X,L)*. Henceforth, I will abbreviate a rule like conta(X) :- member(X,Agent) as conta(X:Agent).

3 I here assume the Principle of Unification, which can be roughly stated as: The information associated with the Mother node is a result of unifying all the relevant pieces of information given in its Daughter nodes. Unification is a series of complex operations involving appending, matching or substitution.

(8)



Again, by unifying the pieces of information given in NP and V, we obtain the information for the Mother node V:

```
(9)  v(SEM)=[conta(X:Agent)]
     Role=Agent :- Case=nom
     Role=[kim]
     Case=nom
```

Here, we first get Role=Agent by resolving the constraint Role=Agent :- Case=nom with an equation Case=nom and then obtain an equation Agent=[kim] on the basis of two equations Role=Agent and Role=[kim]. Finally, we substitute [kim] for Agent in the top formula, thus obtaining:

(10) v(SEM)=[conta(X:[kim])]

This is equivalent to:

(11) v(SEM)=[conta(kim)],

for X:L is defined as 'X:L :- member(X,L)' and X should be a member of the list [kim].

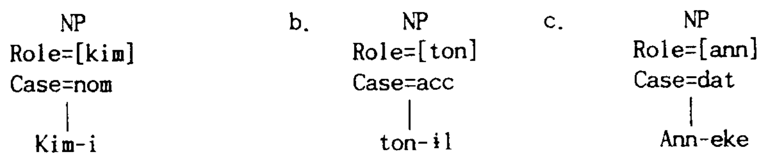
By simply extending the Lexicon, we can treat the free word order illustrated by sentences (3a-f).

(12) Extended Lexicon

```
conta: V, v(SEM)=[conta(X:Agent)], Role=Agent :- Case=nom.
cuəssta: V, v(SEM)=[cuəssta(X:Agent,Y:Recipient,Z:Object)],
                   Role=Agent :- Case=nom,
                   Role=Recipient :- Case=dat,
                   Role=Object :- Case=acc.
Kim: N, Role=[kim].
Ann: N, Role=[ann].
ton: N, Role=[ton].
 -i: CM, Case=nom.
-il: CM, Case=acc.
-eke: CM, Case=dat.
```

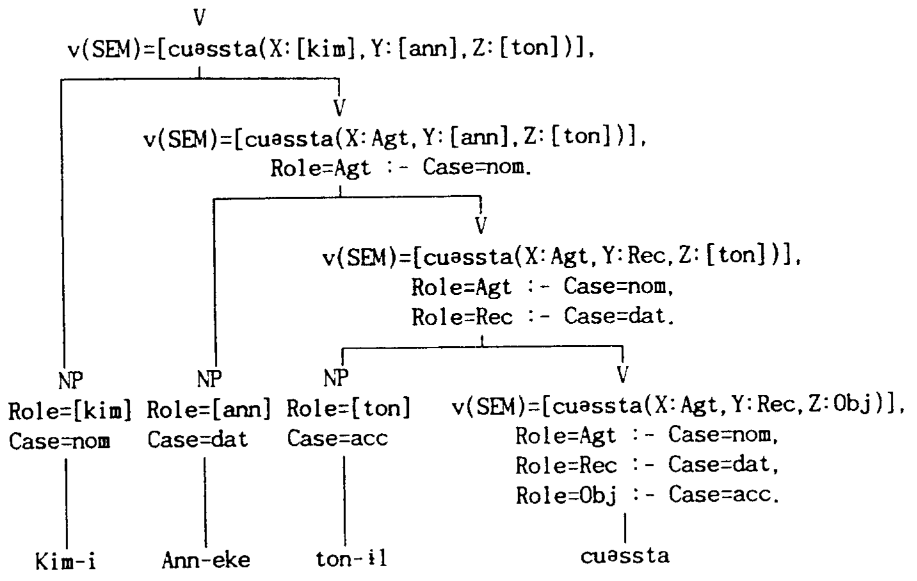
Based on the extended Lexicon, the Case-Marking Rule derives the following trees:

(13) a. b. c.

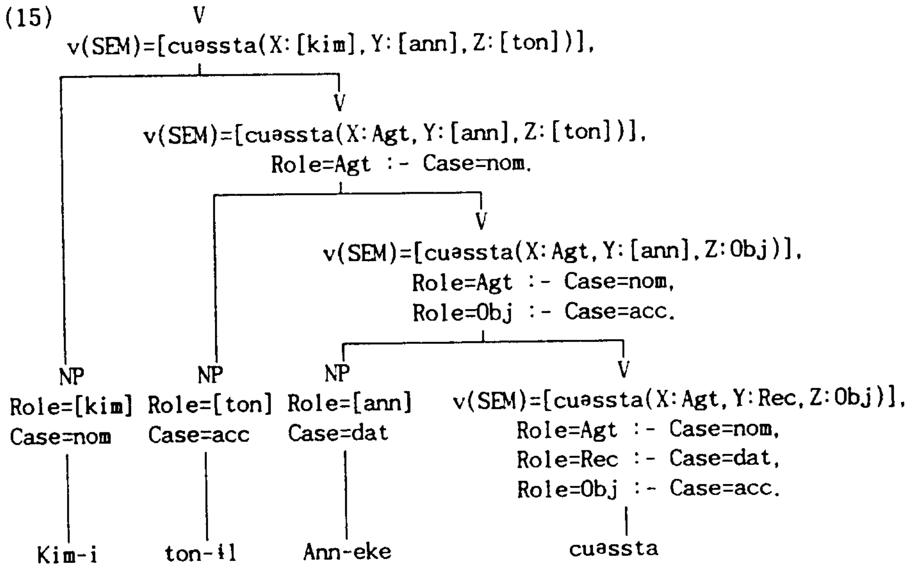


The V-Rule will then derive the following trees:

(14)



(15)



These trees show how sentences (3a) and (3b) can be analyzed. The other trees can be analyzed in the same manner, yielding the same semantic representation. For deriving these trees in which NP's are scrambled, we have only utilized two PS rules and the Lexicon in which each lexical item is provided with appropriate syntactic and semantic information and constraints. The computation of meaning or information required unification and equation solving: the meanings of Daughter nodes were unified and their representations were simplfied by the repeated application of a substitution.

3. Qunatifiers in English

The use of λ-abstraction easily represents the meaning of a quantified NP as a functor:

(16) every man: $\lambda P\forall x[man'(x) \rightarrow P(x)]$

Taking as argument snores'which represents the meaning of a verb 'snores' and undergoing β-reduction, the given functor yields a formula:

(17) a. $\lambda Px\forall[man'(x) \rightarrow P(x)](snores')$
b. $\forall x[man'(x) \rightarrow snores'(x)]$

In this section, the proposed equation solving approach can treat quantifiers equally well. For this, the following annotated PS rules and the Lexicon will be set up.

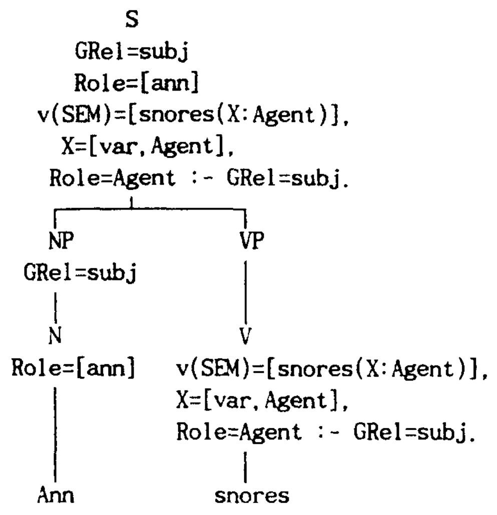
```
(18) Annotated Rules
     S --> NP         VP
           GRel=subj
     NP --> N
     NP --> DET N
     VP --> V
     VP --> V    NP
              GRel=obj

(19) Lexicon
     Ann: N, Role=[ann].
     snores: V, v(SEM)=[snores(X:Agent)],
                       X=[var,Agent],
                   Role=Agent :- GRel=subj.
     loves:  V, v(SEM)=[loves(X:Agent,Y:Patient)],
                        X=[var,Agent],
                        Y=[var,Patient],
                     Role=Agent :- GRel=subj,
                     Role=Patient :- GRel=obj.
     man: N, n(SEM)=[man].
     every: DET, q(SEM)=every([var,Role],P1,P2),
                     P1=n(SEM),
                     P2=v(SEM),
                     Role=[[var,Role]].
```

This small grammar generates sentences like:

(20) a. Ann snores.
     b. Every man snores.
     c. Ann loves every man.

First, (20a) is analyzed as follows:

(21)


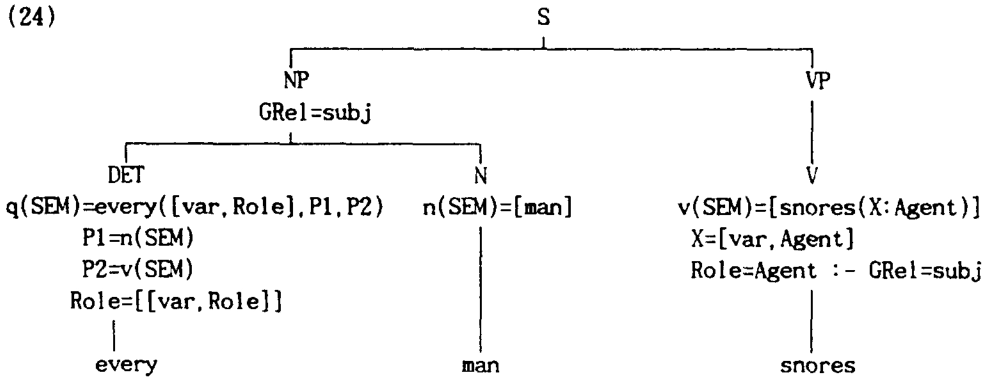
The information structure associated with S simplifies by equation solving to:

(22) v(SEM)=[snores(X:[ann])], X=[var,[ann]].

Since X denotes a variable X such that it ranges over the list [ann], (22) further simplifies to:

(23) v(SEM)=[snores(ann)].

The analysis of sentence (20b) undergoes a similar process of unification and equation solving:

(24)


By unification and substitution, we first obtain:

```
(25)                NP
    q(SEM)=every([var,Role],[man],P2)
              P2=v(SEM)
              Role=[[var,Role]]
                GRel=subj
```

We then obtain the information structure for S:

```
(26)                S
        q(SEM)=every([var,Role],[man],[snores(X:Agent)])
              Role=[[var,Role]]
              X=[var,Agent]
              Role=Agent :- GRel=subj
              GRel=subj
```

Since NP's GRel(ation) is subj(ect), its Role will be Agent. Then by substituting Agent for Role and replacing [var,Agent] with X, we obtain:

```
(27)                S
        q(SEM)=every(X,[man],[snores(X:[X])])
```

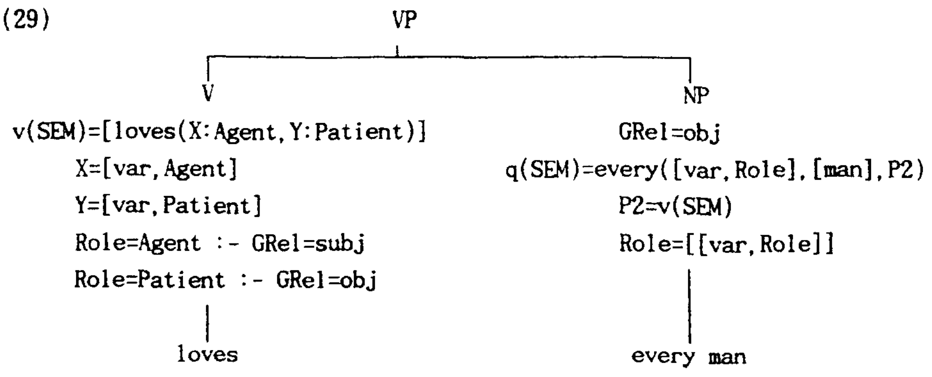
Furthermore, since f(X:[X]) is equivalent to (f(X) :- **member**(X,[X]) and **member**(X,[X]) always holds true, f(X:[X]) simplifies to f(X). Thus, from (27) we get:

```
(28)                S
        q(SEM)=every(X,[man],[snores(X)])4)
```

A quantifier expression may occur in VP as in sentence (20c). But it can also be treated nicely in the proposed grammar. The following is an analysis of the VP 'loves every man':

4 In Prolog, the quantifier every may be treated as a functor defined as: every(X,[f],[g(X)]) :- all(X,f(X) -> g(X)).

(29)



To obtain the information structure for VP, we first unify the two information structures of V and NP and then deduce the fact Role=Patient by resolving the constraint Role=Patient :- GRel=obj and GRel=obj. We now substitute Patient for Role, and Y for [var,Role], [loves(X:Agent,Y:Patient)] for P2 in every([var,Role], [man],P2). As a result of all these operations, we now have:

```
(30)                    VP
        q(SEM)=[every(Y,[man],[loves(X:Agent,Y:Patient)])],
               X=[var,Agent]
               Role=Agent :- GRel=subj
```

When this combines with an NP 'Ann', we obtain:

```
(31)                    S
        q(SEM)=[every(Y,[man],[loves(ann,Y)])]
```

Translated in Prolog, q(SEM) will be:

(32) all(Y,man(Y) -> loves(ann,Y))

We have thus captured the meaning of a universally quantified sentence 'Ann loves every man' through a step-by-step process. The meaning structure of the entire sentence is obtained by unifying parts of the information assigned to each of its phrasal parts. As in Montague Semantics, it is possible to assign a semantic structure to the quantifier 'every' as well as to the quantified NP 'every man'.

4. Conjoined Noun Phrases

It has been pointed out by Moore (1989) and Halvorsen (1991) that the treatment of conjoined NP's like (33) causes a problem for unification-based approaches.

(33) Ann and Bob snore.

But I will show that this conjunction can also be treated nicely in the equation-solving approach. For such an treatment, we just need an NP-Conjunction Rule with the specification of the meaning of 'and' in the Lexicon:

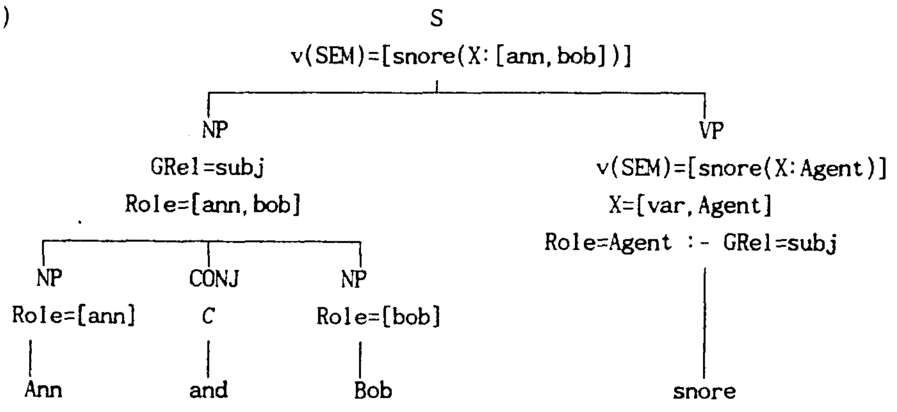
(34) NP --> NP and NP

(35) Lexicon

```
    and: CONJ, MRole=X :- append(DRole1,DRole2,X).
    Ann: N, Role=[ann].
    Bob: N, Role=[bob].
    snore: V, v(SEM)=[snore(X:Agent)],
                     X=[var,Agent],
                 Role=Agent :- GRel=subj.
```

Here in the Lexicon, the constraint assigned to CONJ may be formulated as a general principle on conjunction, stating that MRole, the Role, or semantic content, of the Mother node is the result of appending a DRole, the Role of one of its Daugheter nodes to the other DRole.

(36)



By the contraint *C* on *and*-Conjunction, we first obtain (37) as representing the meaning of the conjoined NP 'Ann and Bob':

(37) Role=[ann,bob]

And then, by unification, we obtain the semantic structure for S:

(38) v(SEM)=[snore(X:[ann,bob])]

The formula given in the list is equivalent to:

(39) a. snore(X) :- member(X,[ann,bob]).
     b. snore(ann), snore(bob).
     c. snore(ann) & snore(bob)

It has thus be shown that a conjoined NP like 'Ann and Bob' can be treated adequately in an equation-solving approach without β-reduction.

I will also show that the proposed analysis of a conjoined NP can apply to the case of a group interpretation.

(40) Ann and Bob met yesterday.

Here, the verb 'met' requires a group as its Subject. By just encoding this information into the semantic structure of the verb, we can obtain the group interpretation of sentence (40).

(41) Addition to the Lexicon

```
met: V, v(SEM) = [met(L:[Agent])],
            L=[var,[Agent]],
         Role=Agent :- GRel=subject.
yesterday: ADV, s(SEM) = [yesterday(v(SEM))].
```

On the basis of this expanded Lexicon, we can construct the following tree:

```
(42)                         S
        s(SEM)=[yesterday(met(L:[[ann,bob]]))]
     ┌───────────────────────┴──────────────────────────┐
    NP                                                  VP
 GRel=subj                   s(SEM)=[yesterday(met(L:[Agent]))]
Role=[ann,bob]                     L=[var,Agent]
     │                        Role=Agent :- GRel=subject.
     │                          ┌───────────────┴──────────────┐
     │                          V                             ADV
     │          v(SEM)=[met(L:[Agent])]   s(SEM)=[yesterday(v(SEM))]
     │          L=[var,[Agent]],                               │
     │          Role=Agent :- GRel=subject.                    │
     │                          │                              │
Ann and Bob                    met                         yesterday
```

The above tree is obtained by a routine process of unification and substitutions, yielding the semantic structure (43) for S.

(43) a. yesterday(met(L:[[ann,bob]]))
     b. yesterday(met([ann,bob]))

Note in (43a) that L is a list [ann,bob]. Hence, (43a) reduces to (43b), as before. (43b) is then interpreted as representing a factual or non-factual information that Ann and Bob as a unit was involved in the meeting of yesterday.

5. Concluding Remarks

One of the motivations for adopting an equation solving approach is to make the process of computing meaning representation more transparent and implementable in a computer language like Prolog. Each information structure is represented by a set of Horn clauses or equations and the composition of informaton structures is carried out by the repeated application of appending, matching or substituion. This process we call equation solving.

There are many problems left unresolved. One of the residual problems has to do with a conjoined NP like:

(44) a. Ann and every man snore.[5]
     b. Bob and every other man snore.

But note here that the set of objects referred to by one conjunct 'Ann' must be disjoint from that referred to by the other conjunct 'every man'. Hence, the following expressions sound inconsistent.

(45) a. *Ann and every female snore.
     b. *The man and every man snore.
     c. *I and I snore.

5 According to the present formulation of the information structure of 'and', sentences like (44a) fail to be treated properly. What is obtained is a wrong representation:
  [i] every(X,[man],[snore(X:[ann,X]),
which is equivalent to:
  [ii] all(X,man(X) -> (member(X,[ann,X]) -> snore(X))).
This does not entail snore(ann). Hence, the analysis is wrong.

On the other hand, the following expressions become acceptable, when they are used to refer to two different persons:

(46) a. This man and this man snored.
b. You and you snored.

A problem like this cannot be resolved by just employing Type Raising or β -reduction. But when this type of constraint is clearly understood, an equation-solving approach can easily encode it into the information structure of a Conjuction 'and'. This I hope to do in the next paper.

REFERENCES

Clocsin, W.F. and Mellish, C.S. (1981) *Programming in Prolog*, Springer-Verlag, Berlin.

Dahl, V. and Patrick, S.(1985) *Natural Language Understading and Logic Programming* North-Holland, Amsterdam.

DeGroot, D. and Gary L.(1986) *Logic Progamming: Fuctions, Relations, and Equations* Prentice-Hall, Englewood Cliffs, New Jersey.

Fenstad, J. et al. (1987) *Situation, Language and Logic*, D. Reidel, Dordrechet.

Fenstad, J. et al. (1990) "Computational Semantics: stpes towards "intelligent" text processing," *Computational Semantics*, COSMOS-Report No. 15.

Gogeun, J.A. (1988) "What is Unification? A Categorical View of Substitution, Equation and Solution," Report No. CSLI-88-124, Stanford.

Halvorsen, P. and Jan T.L. (1991) " Computational Semantics Lectures Slides," LSA Linguistic Institute, Snata Cruz.

Kowalski, R. (1979) *Logic for Problem Solving*, North Holland, New York.

Lee, K. (1989) "The problem of meaning in natural language processing [written in Korean]," *Language Research* (Seoul Nat'l Univ.), 25,1: 225-238.

Lee, K. (1992) "An analysis of some fragments of Korean for computational semantics," *Festschrift for Prof. Tae-Oak Kim*, (in preparation).

Montague, R. (1973) "The proper treatment of quantification in ordinary English." in Hintikka, et al. eds. (1986) *Approaches to Natural Language*, 221-242, D. Reidel, Dordrecht.

Moore, R.C.(1989)"Unification-based semantic interpretation," 27th Annual Meeting of the Association for Computational Linguistics, 33-41.

Pereira, F.C.N. and Stuart M.S. (1987) *Prolog and Natural-Language Analysis*, Center for the Study of Language and Information, Stanford.

Yoo, S. (1989) *Situation Semantics and Extended Categorial Grammar*, Hanshin Publishing Co. Seoul.